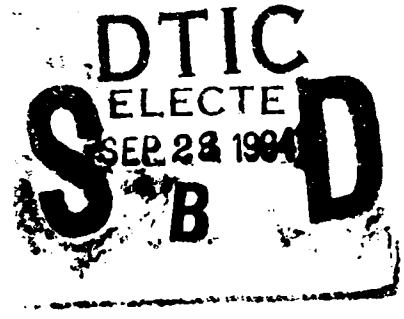
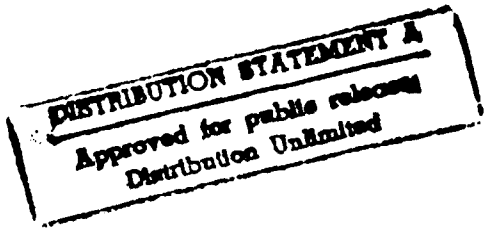
TASK: UU03
CDRL: 05156
March 1993

Reuse Library Framework Version 4.1 Binary Release Installation Guide

Informal Technical Data





STARS-UC-05156/012/00
March 1993

TASK: U03
CDRL: 05156
March 1993

INFORMAL TECHNICAL REPORT

For

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

RLF Binary Release Installation Guide, Version 4.1

STARS-UC-05156/012/00
March 1993

Data Type: A005, Informal Technical Data

CONTRACT NO. F19628-88-D-0031
Delivery Order 0011

Prepared for:

Electronic Systems Center
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

Prepared by:

Paramax Systems Corporation
12010 Sunrise Valley Drive
Reston, VA 22091

TASK: U03
CDRL: 05156
March 1993

Data ID: STARS-UC-05156/012/00

Distribution Statement "A"
per DoD Directive 5230.24
Authorized for public release; Distribution is unlimited.

Developed by: Paramax Systems Corporation

This document, developed under the Software Technology for Adaptable, Reliable Systems (STARS) program, is approved for release under Distribution "A" of the Scientific and Technical Information Program Classification Scheme (DoD Directive 5230.24) unless otherwise indicated. Sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA) under contract F19628-88-D-0031, the STARS program is supported by the military services, SEI, and MITRE, with the U.S. Air Force as the executive contracting agent.

TASK: U03
CDRL: 05156
March 1993

INFORMAL TECHNICAL REPORT
RLF Binary Release Installation Guide, Version 4.1

Principal Author(s):

Del Gordon *Date*

Approvals:

Task Manager *Richard E. Creps* *Date*

(Signatures on File)

TASK: U03
CDRL: 05156
March 1993

INFORMAL TECHNICAL REPORT
RLF Binary Release Installation Guide, Version 4.1

Change Record:

Data ID	*Description of Change*	*Date*	*Approval*
STARS-UC-05156/012/00	Updated for PCTE integration enhancements.	March 1993	*on file*
STARS-UC-05156/002/00	Original Issue	November 1992	*on file*

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Informal Technical Report

4. TITLE AND SUBTITLE

RLF Binary Release Installation Guide

5. FUNDING NUMBERS

F19628-88-D-0031

6. AUTHOR(S)

Paramax Corporation

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Paramax Corporation
1210 Sunrise Valley Drive
Reston, VA 22090

8. PERFORMING ORGANIZATION REPORT NUMBER

STARS-UC-05156/012/00

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Air Force
Headquarter, Electronic Systems
Hanscom AFB, MA 01731-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

05156

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution "A"

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This manual tells you how to install, build, and start up the *Binary Release* of the Software Technology for Adaptable and Reliable Systems (STARS) Reuse Library Framework (RLF) and its graphical user interface, the Graphical Browser (GB), hereafter referred to as the RLF GB. The RLF system includes the *Library_Manager* application.

14. SUBJECT TERMS

15. NUMBER OF PAGES

45

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unlcassified	Unclassified	Unclassified	SAR

Contents

List of Figures

1 Introduction

1.1 Scope

This manual tells you how to install, build, and start up the *Binary Release* of the Software Technology for Adaptable and Reliable Systems (STARS) Reuse Library Framework (RLF) and its graphical user interface, the Graphical Browser (GB), hereafter referred to as the RLF GB. The RLF system includes the `Library_Manager` application. This version of RLF
Δ can support execution using Portable Common Tools Environment (PCTE) as an underlying object management system. If you run the PCTE version, then certain guidelines must be followed when installing and running the software. PCTE-specific information has been gathered in Appendix B.

You should read these instructions before you install the RLF software. The reader is expected to be a system administrator, or fulfill that role; therefore, familiarity with the UNIX C shell, UNIX files and directories, and basic X Window System (X) knowledge is assumed. Since this version of RLF can support execution using Emeraude's implementation of PCTE as an underlying object management system, if the RLF is run with PCTE, it is assumed that the user understands PCTE and the Emeraude product, including the ability to log in to the PCTE environment, and construct and run *esh* scripts.

1.2 Other Documents

If you want to install the *source code release* of the RLF, then you should read the *RLF Source Code Release Installation Guide.* The source code release does not contain executable programs; they must be built from the source code.

Other RLF documents exist that contain more detailed information about different aspects of the RLF. For information on constructing RLF libraries, consult the *RLF Modeler's Manual.* For detailed information on RLF administration issues, see the *RLF Administrator's Manual.* For help with use of the RLF and the RLF GB, see the *RLF User's Manual.*

1.3 Overview of Document

The *RLF Binary Release Installation Guide* contains the following sections:

- Section 2, *Installing the Binary Release of the RLF Software*, describes how to load the executable RLF software onto your system from the distribution tape or from the Internet. The procedures for configuring and installing the RLF binary release are then described.

  The RLF binary release contains pre-built, executable software, and does not contain any source code.

- Section 3, *Verifying the RLF Installation*, describes the procedures for starting up the RLF Graphical Browser in order to verify that the installation was successful.

1.4 Identification

This manual is for the RLF Binary Release, Version 4.1, March, 1993. This release runs on SunOS, version 4.1.1 or later.

The only other software dependency that exists is the following:

- MIT X11R4

Δ The PCTE version of the RLF software depends on the above software at **run-time**, plus:

Δ - Emeraude PCTE, v.12.3

1.5 Product Overview and Rationale

Within a particular problem domain, applications tend to have similar characteristics. To facilitate reuse within a domain, it is useful to analyze the domain to identify the similarities and differences among applications, and to record the results of this analysis in the form of a domain model and generic domain architecture. A crucial requirement for domain-specific reuse libraries of the future is that they provide a means for storing and accessing this domain knowledge to readily support reuse-based application development.

The RLF was designed specifically to meet this need. The RLF supports the development and evolution of comprehensive domain models consisting of taxonomic information to describe domain structure and rule bases to capture heuristic domain knowledge. Such a domain model is capable of supporting many tools that impact all phases of the software life cycle and support reuse of a broad spectrum of life cycle products beyond just source code.

The search and retrieval mechanisms of the RLF provide both novice and experienced users effective reuse library access. The RLF provides several modes of user interaction to support different search strategies. The fundamental mode of operation relies on a browsing paradigm, wherein users are presented with a graphical representation of the library domain model and move through the library under their own control, deciding which kinds of assets to investigate. The RLF also provides an advisor mode, which gives users rule-based advice about which assets to investigate next, and a query mode which allows users to submit database-style queries about library assets.

The main purpose of the RLF GB is to provide a graphical user interface to the RLF. This graphical interface is the primary user interface when the RLF is used as a domain-specific

software reuse library tool. Another purpose of the RLF GB is to demonstrate large-scale Ada software reuse. The RLF GB itself reuses the RLF and RGB software subsystem components.

The RLF GB provides to the user a graphical, point-and-click interface to the RLF and its structured domain knowledge. The graphical interface to the RLF has completely replaced the textual interface of older versions of the RLF. The `Library_Editor` application has become the **`Library_Manager`** and also has a graphical interface; the textual interface is no longer supported. The RLF GB relies on the X Window System (X) to provide some of its capabilities, such as windowing, pull-down menus, graphics display, and mouse control. With these capabilities the RLF GB can graphically depict an RLF library, and the user can view and interact with the library using simple mouse actions. The library structure is also referred to as a *domain model.*

The RLF GB provides read-only access to the domain model; it does not provide any network creation or modification capabilities. The **`Library_Manager`** application provides some library modification capabilities, but is only discussed in this manual in relation to its installation. For a detailed discussion of the **`Library_Manager`**, see the *RLF Administrator's Manual.*

1.6 Notation Used in this Manual

This manual describes procedures to interact with the UNIX operating system through a C shell interpreter. Therefore, examples are given that sometimes use a special notation. In presenting the examples, this document uses the following notational conventions:

- `typewriter font`

  This font represents information displayed by the computer. It is also used in code examples and textual passages to indicate use of the C shell command language or names of UNIX or application programs.

- *italic font*

  This font is used in textual passages and code examples to indicate user-specified parameters for program names or command line options; it is also used to indicate special terms or phrases used in textual descriptions.

- `boxed typewriter font`

  Boxed typewriter text indicates information that you type **exactly as shown** as input to the computer.

- *boxed italic font*

  Boxed italic text indicates information that **you must supply** and type as input to the computer.

- **%**

  The percent sign is used to represent the C shell prompt.

- **[Quit]**

  This bold font, enclosed in square brackets, represents a graphical command button, or menu option that is available from a graphical pull-down menu or cascading menu.

1.6.1 Notation Examples

In the following example you would type the text "`source $HOME/.cshrc`" but you would not type the percent sign ("**%**"), which is the C shell prompt; also note that your C shell prompt may appear differently:

```
% source $HOME/.cshrc
```

In the next example you would insert the name of a text editor of your choice, such as `vi` or **`emacs`**, but you would type the filename "`.cshrc`" as shown:

% *editor* `$HOME/.cshrc`

The following example shows the use of the square brackets to indicate a graphical command button. These buttons are displayed by X and must be clicked on with a mouse to be activated:

Select the **[Quit]** button to exit the RLF GB.

2 Installing the Binary Release of the RLF Software

This section contains the instructions to install the binary release of the RLF.

2.1 Installation Overview

The following list summarizes the steps required to install the Binary Release of the RLF. This section is only an overview; each installation step is discussed in more detail in the following sections.

1. Check installation requirements:
   - Hardware requirements
   - Memory requirements (main memory, disk space, swap space)
   - Software requirements
2. Install RLF software from magnetic tape or download from network, using one of the following two procedures:
   - Load software from tape, as follows:
     - Make sure you have the correct distribution tape.
     - Choose a tape drive.
     - Create a directory to store the software.
     - Load the software from tape.

   —**or**—
   - Download software from network, as follows:
     - Create a directory to store the downloaded software.
     - Download the software from the network.
3. Extract and install RLF software from the archive file:
   - Extract the software from the archive file.
   - Configure and install the software.
4. Verify the installation.

2.2 Checking Installation Requirements

Make sure your site configuration meets all the requirements listed in this section before you try to install and run the RLF Binary Release.

2.2.1 Hardware Requirements

The RLF software requires the following type of workstation:

- Sun-4 workstation (SPARCstation)

The smallest Sun workstation we recommend running RLF on is as follows:

SPARC IPC with 16 megabytes (MB) of memory and 60 MB of swap space

The important factor in running an RLF application is the amount of physical memory available along with the amount of swap space available (given that you are running on a SPARC chip).

There are many different display devices (also known as "framebuffers") available for Sun workstations. The display device for your workstation must be supported by the windowing system you use.

2.2.2 Memory Requirements

To run the RLF software, you must meet the following memory requirements:

- You must have at least 8 MB of main memory. For improved performance, increase memory size—16 MB of main memory is recommended, and 32 MB is preferred.
- You must have at least 40 MB of disk space to install the RLF Binary Release software.
- You must have at least 60 megabytes of swap space to run the RLF software with the Graphical Browser. (It is recommended that if you have more than 60 megabytes available to allocate to swap space that you do so. Anywhere from 60 to 120 megabytes of swap space is recommended.)

Use the **df** command to check how much disk space you have. The "avail" column shows the amount of free disk space (in thousands of bytes).

```
% df
```

Use the **pstat** command to check how much swap space you have:

```
% /etc/pstat -T
```

The second number on the last line of the output is the swap space (in thousands of bytes).

2.2.3 Software Requirements—Run-Time

The RLF software depends on the following software at **run-time**:

- This manual assumes a UNIX C shell interpreter is accessible to the user.
- The MIT X Window System, Release 4
- SunOS Release 4.1.1 or later

Δ The PCTE version of the RLF software depends on the above software at **run-time**, plus:

Δ
- Emeraude PCTE, v.12.3

To check which version of the operating system your site is running, use the **head** command to print out the first line of the message of the day:

```
% head -1 /etc/motd
```

The RLF GB can be executed on any workstation running X11R4. This can be the Sun-supplied implementation of X, OpenWindows, or the X distribution from MIT.

There are a number of window managers that one may use in an X environment. The RLF GB has been used successfully under the following such window managers: **mwm** (Motif Window Manager), **twm** (the official MIT-distributed window manager), and the Visual User Environment (VUE) (from SAIC or Hewlett-Packard).

2.3 Installing the RLF Binary Release from Tape

This section explains how to load the RLF Binary Release from the distribution tape.

To load the RLF Binary Release from tape, you must:

1. Make sure you have the correct distribution tape (or tapes).
2. Choose a tape drive (either the one on your local machine or one on another machine on your network).
3. Extract the RLF software from the tape, by following the instructions described below.

2.3.1 The RLF Distribution Tape

The RLF software is distributed on tape in `tar` format. There are different versions of the distribution tape, depending on the type of release being installed: binary or source code.

The machine architecture supported is Sun-4 (SPARC). This is the only architecture supported at this time. The Sun-3 architecture is no longer supported.

You can use the `arch` command to verify what type of machine architecture you have:

```
% arch
```

and the output should be:

```
sun4
```

2.3.2 Choosing a Tape Drive

For tape installation, this guide discusses local installation only, however, you can perform installation from tape in two ways:

Locally *Local* installation occurs when you use the tape drive of the machine on which the software is to reside.

Remotely *Remote* installation occurs when you use the tape drive of a machine other than the one on which the software is to reside; the machine whose tape drive you use (the *remote* machine) must have a network connection to the machine on which the software is to reside (the *local* machine). In this case, you must be able to access the remote machine by using the `rsh` command.

2.3.3 Extracting the RLF Binary Release from the Tape

To extract the RLF software from the tape, use the `tar` command. To successfully complete the extraction procedure, you must have the following information available:

1. The pathname of the directory where you plan to install the RLF software (also, make sure you have write access to this directory). This directory will be referred to as the RLF "home" directory, and its value will be denoted by the environment variable `RLFHOME` in the following examples, or `$RLFHOME` when referencing the variable.
2. The name of the tape device (e.g., `/dev/rst1`).

2.3.4 Summary of Binary Release Installation Steps

1. Extract the files on the tape into the RLF home directory.

   % `setenv RLFHOME` *directory_name*

   % `cd $RLFHOME`

   % `tar xvf` *tape_device*

2. Run the `Install_RLF_bin` script.

   % `Install_RLF_bin`

   —or for PCTE:

   % `Install_RLF_pcte_bin`

   - Run the installation script interactively, or
   - Edit the configuration file and run the installation script in batch mode

The RLF installation script is described in further detail on Section 2.6.

2.4 Obtaining the RLF Binary Release via Anonymous FTP

This section explains how to obtain the RLF Binary Release via anonymous FTP over the Internet, using the UNIX `ftp` program. (FTP is an acronym that stands for the "File Tranfer Protocol." The UNIX program `ftp` is an implementation of the FTP protocol.) Once the necessary archive files have been transferred to your local host, you must extract the RLF files from the archive. This section also provides instructions for this operation.

To access the RLF software via anonymous FTP, you must:

1. Choose a local directory where the RLF archive file will be downloaded to.
2. Establish an anonymous FTP connection over the Internet.
3. Transfer the appropriate files.
4. Extract the RLF software from the archive file, by following the instructions described below.

First choose a directory where the RLF archive files will be stored. Then set your current working directory to that directory.

For example:

```
% cd directory_name
```

Now establish a connection to the STARS anonymous FTP host. The Internet name of this host is: `falcon.stars.rosslyn.paramax.com`. To establish an FTP connection, issue the `ftp` command with a host name as an argument. For example:

```
% ftp falcon.stars.rosslyn.paramax.com
```

The following is a sample FTP session. The text you must type is enclosed in boxes. The FTP prompt is indicated by the string "`ftp>`".

```
Connected to falcon.stars.rosslyn.paramax.com.
220 falcon FTP server (Version 5.107 ) ready.
Name (falcon.stars.rosslyn.paramax.com:username): anonymous
331 Guest login ok, send ident as password.
Password: your_network_address
230 Guest login ok, access restrictions apply.
ftp> cd pub/RLF
250 CWD command successful.
ftp> ls -al
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 5706
drwxr-xr-x 2 103 512 Jun 29 15:39 .
drwxrwxrwx 10 ftp 1024 Aug 14 19:00 ..
-rw-r--r-- 1 116 491040 Nov 30 1992 RLF_4.1_bin.tar.Z.split-aa
-rw-r--r-- 1 116 488439 Nov 30 1992 RLF_4.1_bin.tar.Z.split-ab
-rw-r--r-- 1 116 479725 Nov 30 1992 RLF_4.1_bin.tar.Z.split-ac
-rw-r--r-- 1 116 495984 Nov 30 1992 RLF_4.1_bin.tar.Z.split-ad
-rw-r--r-- 1 116 400538 Nov 30 1992 RLF_4.1_bin.tar.Z.split-ae
-rw-r--r-- 1 116 461326 Nov 30 1992 RLF_4.1_bin.tar.Z.split-af
-rw-r--r-- 1 116 507647 Nov 30 1992 RLF_4.1_bin.tar.Z.split-ag
226 Transfer complete.
remote: -al
1006 bytes received in 0.23 seconds (4.2 Kbytes/s)
ftp> binary
200 Type set to I.
ftp> mget RLF_4.1_bin*
mget RLF_4.1_bin.tar.Z.split-aa? y
200 PORT command successful.
150 Opening ASCII mode data connection for RLF_4.1_bin.tar.Z.split-aa
(491040 bytes)
.
```

```
226 Transfer complete.
local: RLF_4.1_bin.tar.Z.split-aa remote: RLF_4.1_bin.tar.Z.split-aa
493040 bytes received in 75 seconds (6.4 Kbytes/s)
mget RLF_4.1_bin.tar.Z.split-ab? y
200 PORT command successful.
150 Opening ASCII mode data connection for RLF_4.1_bin.tar.Z.split-ab
(488439 bytes)
.
226 Transfer complete.
local: RLF_4.1_bin.tar.Z.split-ab remote: RLF_4.1_bin.tar.Z.split-ab
490439 bytes received in 75 seconds (6.4 Kbytes/s)
mget RLF_4.1_bin.tar.Z.split-ac? y
200 PORT command successful.
150 Opening ASCII mode data connection for RLF_4.1_bin.tar.Z.split-ac
(479725 bytes)
.
```

(Answer yes (y) to all the **mget** prompts.)

⋮

```
ftp> quit
221 Goodbye.
```

All the RLF archive files should now have been transferred to your local directory. The next section describes the procedure to extract the RLF software from the archive files.

2.5 Extracting the RLF Binary Release from an Archive File

The RLF archive files residing on the anonymous FTP host are in UNIX **tar** format (*tar* stands for *tape archive*, but a tape archive can also exist on a disk). In addition, the **tar** files have been **split** and **compressed**. The UNIX **split** command divides a large file into many smaller files. The UNIX **compress** command makes files smaller in size through the use of a file compression algorithm.

The intent of the extraction process is essentially to "undo" what the **tar**, **compress**, and **split** programs have done to the RLF software. Therefore, what needs to be done is to concatenate all the split files back together, uncompress them, and extract the RLF software files from the resulting archive.

When you extract the RLF software from the archive, the **tar** command will create the necessary directory hierarchy automatically, starting at the current directory. Therefore, *before you start the extraction process, ensure that you are in the desired directory.*

To extract the RLF software from the archive files, issue the following command:

```
% zcat RLF_4.1_bin.tar.Z.split* | tar xvf -
```

The **zcat** command uncompresses and concatenates the split files together, and the results are then piped (the vertical bar "|" represents a UNIX pipe) to the **tar** command which extracts the files from the archive. The dash ("-") at the end of the **tar** command is important—it tells the **tar** program to read its input from the pipe instead of from a file, so be sure to include it in your command line.

When this command is complete, you should have a complete directory structure, starting at your current working directory, of all the RLF executable software, and documentation. The RLF installation script, **Install_RLF_bin**, should now reside in this directory. (The PCTE installation script is named **Install_RLF_pcte_bin**).

The next step is to run the RLF Binary Release installation script. This script is described in further detail in the following section.

2.6 Configuring and Installing the RLF Binary Release

For the Binary Release, the RLF installation script performs the following tasks:

- ensures configuration variables are properly set
- installs sample RLF libraries
- copies the **RLF_Browser** file to **/usr/lib/X11/app-defaults**
- creates the **bitmaps** directory in **/usr/lib/X11/app-defaults/bitmaps** and copies the bitmap files into that directory

There are two ways to run the installation script: interactively, or in batch mode. The interactive mode prompts you for all the necessary information. In batch mode, you edit the file **Install_Rlf.var** and supply all the necessary information by setting environment variables appropriately.

Since the **Install_RLF_bin** script copies files into system directories, you must be "root" or have root privileges to successfully run the script. To complete the installation of the RLF software, log in as "root" if you're not already, and invoke the **Install_RLF_bin** script:

```
% su
Password: root_password
% Install_RLF_bin
```

—or for PCTE:

```
% Install_RLF_pcte_bin
```

The installation script assumes the existence of the directory:

```
/usr/lib/X11
```

and

```
/usr/lib/X11/app-defaults
```

for installing X-related data files. However, if it is not feasible or desirable to install the "RLF_Browser" file or bitmap files in the above directories at your site, then consult Appendix A, "Customization," of this document for alternative approaches.

2.6.1 Running the Installation Script Interactively

If you choose to run the installation script interactively, you will be prompted for all the necessary site-specific information. A sample run of the installation script is given below.

This example assumes you have set the environment variable RLFHOME to an appropriate value before issuing the following commands.

```
% cd $RLFHOME
% Install_RLF_bin
```

—or for PCTE:

```
% Install_RLF_pcte_bin
```

```
+-------------------------------------------------------------+
|                                                             |
|                RLF 4.1 Installation Script                  |
|                      Binary Release                         |
|                                                             |
```

```
+-------------------------------------------------------+

   You must choose one of the following installation options:


      1.) Interactive installation

            * You are prompted for all necessary
              configuration values (i.e., pathnames).


      2.) Edit the file that contains the configuration values

            * You edit the file "Install_Rlf.var" and
              set the configuration values appropriately
              for your site.


      3.) EXIT this script.


      (If you do not edit the "Install_Rlf.var" file, or specify
      invalid values, you will be prompted for the configuration
      values anyway.)


      Which installation option do you prefer?


Please enter 1, 2, or 3 > [1]


You chose: 1

+-----------------------------------------------+
|  Executing interactive installation script.   |
+-----------------------------------------------+

Define the site-dependent environment variables.
-------------------------------------------------
```

NOTE: If you have already set the RLFHOME variable, then the fol-

lowing section of the installation script will not be executed.

```
Specify path to top-level RLFHOME directory
-------------------------------------------
  Examples:
        /mybase/RLF
        /afs/myhome/see/rlf
        /usr/tools/rlf
        etc.


RLFHOME = /afs/myhome/see/rlf
```

NOTE: If you have already set the RLFHOME variable, then the above section of the installation script will not be executed.

```
     RLFHOME = /afs/myhome/see/rlf
      RLFBIN = /afs/myhome/see/rlf/bin

  APPDEFAULTS = /usr/lib/X11/app-defaults

Moving the RLF GB resource file (RLF_Browser) to /usr/lib/X11/app-defaults

Moving the RLF GB bitmap files to /usr/lib/X11/app-defaults/bitmaps

Installation Complete
```

The installation of the RLF system is now complete. By issuing an *ls* command, you should be able to determine that the `bin` subdirectory of the **RLFHOME** directory contains the following executable programs:

```
% ls $RLFHOME/bin
```

```
.rlfrc
Graphical_Browser
Library_Manager
Lmdl
```

```
RLF_Browser
RLF_GB
Rbdl
Sndl_to_Lmdl
bitmaps (a directory)
less
view_stp.csh
xloadimage
```

2.6.2 Running the Installation Script in Batch Mode

To run the installation script in batch mode, you must first edit the file `$RLFHOME/code/Install_Rlf.v` and edit the following section of that file; there is one environment variable that must be set appropriately for your site:

```
#setenv RLFHOME        /afs/myhome/test/rlf_4.1
```

Note that all the above line is commented out (the pound-sign (#) is the C shell symbol that indicates the rest of the line is to be ignored). You must "un-comment" out this line (i.e., delete the pound-sign) and supply the appropriate values for all the given variables.

If you supply erroneous values for any of the variables, then the script will notify you during its execution because it checks the validity of all supplied variables. The script will prompt you for any variables that are not set or that are set improperly.

The type of data that must be supplied for each variable is described in the following list:

RLFHOME *Pathname*
Supply a pathname where the entire RLF directory hierarchy will reside.

The following environment variables are given the default values shown below. If these values are invalid for your site, then you must supply valid values or else the build will fail:

APPDEFAULTS `/usr/lib/X11/app-defaults`
The standard pathname of the X11 directory for system-wide application defaults files.

2.6.3 On-Line "man" Pages Installation Procedures

This section describes the procedures for installing and using the RLF "man" pages (an abbreviation of "manual" pages). The UNIX system maintains the convention of providing

on-line reference manual pages for programs. Using the UNIX *man* command, a user may display information from any "man" pages that have been installed on the system and that are in the "man" page search path.

The "man" page sources are conventionally located in the **/usr/man** directory. Typically there are subdirectories in **/usr/man** such as **man1**, **man2**, **man3**, etc., that refer to major sections of the UNIX reference manuals, and possibly additional subdirectories such as **manl** that have been tailored for your site. For example, **manl** refers to the **local** "man" pages.

The procedures described below assume that the RLF "man" pages have been installed in the **$RLFHOME/man/man1** subdirectory, and that they will remain in that location. If desired, you may copy or move the **$RLFHOME/man/man1** directory to the standard **/usr/man/man1** directory or some other directory where "man" pages are located at your site.

Procedure to install RLF "man" pages in the $RLFHOME/man directory:

Create preformatted versions of the RLF "man" pages from the input files residing in **$RLFHOME/man/man1** and place them into the newly created directory **$RLFHOME/man/cat1**, and then create a small "whatis" database file in **$RLFHOME/man**:

```
% /usr/etc/catman -M $RLFHOME/man 1
```

Edit your **$HOME/.login** or **$HOME/.cshrc** file, adding the line:

```
setenv MANPATH "/usr/man:$RLFHOME/man"
```

Note that you may have to alter the above **MANPATH** environment variable specification slightly if you are already setting **MANPATH**, such as to include local man pages:

```
setenv MANPATH "/usr/man:/usr/local/man:$RLFHOME/man"
```

To activate the **MANPATH** environment variable, either **source** the **.login** (or **.cshrc**) file that you edited, or log out and then log back in:

```
% source $HOME/.login   —or—   % source $HOME/.cshrc
```

2.6.4 User-Specific Installation Procedures

These are the items that specific users may want to consider:

- `.rlfrc`
  The `.rlfrc` file is the RLF initialization file.
- `RLF_Browser`
  The `RLF_Browser` file is the X application resource file.
- `MANPATH`
  The `MANPATH` environment variable.

The RLF looks for the `.rlfrc` file in the user's home directory (`$HOME`) or the current working directory. If the `.rlfrc` file is found there, then the RLF reads in initialization data from the file. Command-line options override any settings in the `.rlfrc` file.

The `RLF_Browser` file is used by the X Window System to configure the application according to user- or site-specific parameters. The user may place a copy of the `RLF_Browser` file into his or her home directory (`$HOME`) and then modify it to suit individual tastes. A `RLF_Browser` file in the user's home directory will automatically have higher precedence than the `RLF_Browser` file in `/usr/lib/X11/app-defaults`. Alternatively, the user may place the `RLF_Browser` file in any directory of their choice, and then set the `XAPPLRESDIR` environment variable to that pathname, as in the following example:

```
% cp /usr/lib/X11/app-defaults/RLF_Browser /home/mydir/tools/rlf

% setenv XAPPLRESDIR /home/mydir/tools/rlf
```

The above example assumes the user is able to place a copy of the `RLF_Browser` file into the directory */home/mydir/tools/rlf.*

Also, users may want to edit their `$HOME/.login` or `$HOME/.cshrc` file, adding the line:

```
setenv MANPATH "/usr/man:$RLFHOME/man"
```

so that they may access the RLF "man" pages.

3 Verifying the RLF Installation

This section describes procedures to verify that the RLF installation is correct. The installation of the RLF software is considered to be correct if the RLF GB can be invoked successfully and an RLF library browsed.

3.1 Initial Setup Procedures

This manual assumes that you know how to start up the X Window System on your workstation. For more detailed information on starting and customizing X, see the X document listed in Appendix D, "References."

If the RLF GB has been built, then there will be an executable file named **Graphical_Browser** available for execution. Place the pathname where the **Graphical_Browser** file resides into your command search path. This is accomplished by modifying the value of the C shell **path** variable, either temporarily or permanently.

You can temporarily place the pathname of the **Graphical_Browser** executable into your command search path by modifying the C shell **path** variable. This can be done by issuing the following commands at your C shell prompt:

```
% setenv RLFHOME rlf_pathname
% set path = ( $RLFHOME/unix/bin $path )
```

—or for PCTE:

```
% set path = ( $RLFHOME/pcte/bin $path )
```

This will add **$RLFHOME/unix/bin** or **$RLFHOME/pcte/bin** to your current command search path, where *rlf_pathname* is a pathname you supply. However, when your current shell is exited, this modification will be lost.

You can permanently place the pathname of the **Graphical_Browser** executable into your command search path by editing the C shell initialization file (named **.cshrc**) in your home directory and inserting the **set path** command, as in the following:

```
% edit $HOME/.cshrc
```

insert the following line in the **.cshrc** file:

```
set path = ( $RLFHOME/unix/bin $path )
```

—or for PCTE:

```
set path = ( $RLFHOME/pcte/bin $path )
```

after any existing **set path** commands, where **$path** is the shell variable that contains your current command search path, and **$RLFHOME** is the previously set environment variable that specifies the RLF home directory. The **unix/bin** or **pcte/bin** directory directly beneath **$RLFHOME** contains the **Graphical_Browser** executable. Alternatively, you can add **$RLFHOME/unix/bin** or **pcte/bin** to any **set path** statement that may already exist in your **.cshrc** file. Since the C shell reads the **.cshrc** file every time it is started, these variable settings are effectively permanent until you edit the **.cshrc** again.

There are two environment variables that must be set before the RLF GB can be run successfully. They are as follows:

- DISPLAY
- RLF_LIBRARIES

You can edit the **.cshrc** file so that these variables are automatically set every time you invoke a new C shell.

3.1.1 The DISPLAY Environment Variable

The DISPLAY environment variable is used by X to determine the host where your X server is running, as well as the number of the display to be used on that host. To have this environment variable set automatically each time you log in, insert the following line into your **.cshrc** file:

```
% edit $HOME/.cshrc
% setenv DISPLAY hostname:0
```

where the *hostname* is the name of your computer system as it is known to your network, and ":0" refers to the first display screen of your system (most conventional computer systems are of the one-display type, but X allows applications to run on multi-display systems). You can obtain your host name, if you do not already know it, by issuing the UNIX command **hostname**.

For example, if you issued the **hostname** command, as follows:

```
% hostname
```

and you received the following output:

```
sparc10
```

then you would set your DISPLAY environment variable as follows:

```
% setenv DISPLAY sparc10:0
```

3.1.2 The RLF_LIBRARIES Environment Variable

The RLF_LIBRARIES environment variable is used by the RLF to determine the location of the RLF libraries to be read. An RLF library must be created before the RLF GB can be used to browse that library. However, the UNIX Binary Release of the RLF v.4.1 contains pre-translated sample libraries that are ready to browse in the **$RLFHOME/Libraries** directory. (The PCTE release does not contain pre-translated libraries—see Appendix B for details.) The pre-translated sample libraries that are provided are as follows:

- Demo Actions
- Sort and Search Algorithms

The procedures to create the other RLF sample libraries that are provided with the release in their source form are explained in detail in the *RLF Modeler's Manual*, listed in Appendix D, "References." However, a brief description of these procedures is also provided in this document for the purpose of completely verifying the installation. See Section 3.2.2 for a description of library model specification translation procedures.

To set the RLF_LIBRARIES environment variable to the location of the sample RLF libraries provided with this release, issue the following command:

```
setenv RLF_LIBRARIES $RLFHOME/Libraries
```

To have the RLF_LIBRARIES environment variable set automatically each time you log in, insert the following line into your **.cshrc** file:

```
setenv RLF_LIBRARIES rlf_library_pathname
```

where *rlf_library_pathname* is the pathname to a directory that contains at least one previously created RLF library.

The `source` command is a valuable tool for working in your current shell environment. When you execute the `source` command, your C shell reads and executes the commands in the specified file. Since no new subshell is created, you can use `source` to modify your current environment. Therefore, after editing your `.cshrc` file, you can then "`source`" it so that the new environment variables will be read by your current shell. This has an effect similar to exiting your current C shell and starting a new C shell. To "`source`" your `.cshrc` file, type the following command:

```
% source $HOME/.cshrc
```

where the UNIX environment variable `$HOME` is replaced by the C shell with the path to your home directory.

If you need more information about setting up your X environment, consult the X documentation listed in Section D. Experienced users may wish to refer to Appendix A, "Customization."

3.2 Invoking the RLF GB

After the setup procedures have been completed, the RLF GB can then be started. A C shell script is provided for this purpose, or, the `Graphical_Browser` can be invoked directly. The C shell is provided to check the values of the environment variables and the status of files that the RLF GB reads during execution. The RLF GB can be invoked without the use of this startup script; it is entirely optional. The RLF GB startup script is provided as a convenience to inexperienced RLF users in an attempt to automate some of the status checking that would be performed in a trouble-shooting session.

The startup script is named `RLF_GB`. To invoke the RLF GB, execute the `RLF_GB` script by typing its name at the C shell prompt and pressing the Return key:

```
% RLF_GB
```

After invoking the RLF GB you should see the Main Menu appear in a new window (see Figure 1). Operating procedures for the RLF GB are explained in further detail in the following sections.

To invoke the RLF GB directly, without using the RLF GB startup script, type the name of the program on the command line:

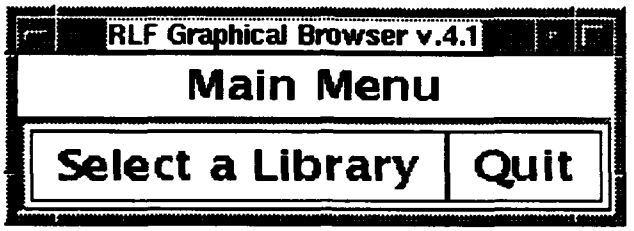


Figure 1: The Main Menu

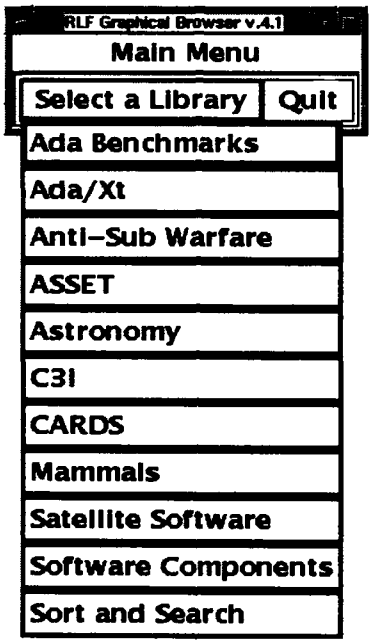


Figure 2: A Sample Select a Library Menu

```
% Graphical_Browser
```

As with the startup script, after invoking the RLF GB you should see the Main Menu appear in a new window.

3.2.1 Browsing an RLF Library

The final verification that the browser portion of the RLF installation was successful is achieved when you can browse an RLF library. You can select a library to browse by clicking the mouse pointer on the **Select a Library** option of the Main Menu. This will invoke a pull-down menu, similar to the one shown in Figure 2.

You can select a library to browse by positioning the screen pointer over a library name and clicking the first or second (usually the left or middle) mouse button.

A pull-down menu, such as the Select a Library menu, will stay posted on the screen after it is invoked with a mouse click. You can then select one of the options available in the menu by moving the pointer over the desired option and clicking the mouse on that button. For now, choose the library named, "Sort and Search" by placing the pointer inside the box labeled "Sort and Search" and clicking the first or second mouse button.

After you select a particular library, the RLF GB displays some informational statements regarding its processing status in the original window from which the application was invoked. These statements reflect the progress of the RLF GB's internal processing and can be safely ignored. They are displayed because large RLF libraries that contain hundreds of concepts can take a minute or more to process; therefore, they indicate that the program is working properly. If the RLF GB does encounter a problem with its input or output processing, an exception will be raised and an error message will be displayed. A list of error messages and a description of their possible causes is given in an appendix of the *RLF User's Manual.*

The status information appears similar to the following:

```
Welcome to the RLF Graphical Browser.
          Version 4.1

Copyright 1992, 1993 Paramax Systems Corp.


Main browser loop...


Getting the root node of the RLF network.
Loading the current state of the RLF network.
Creating the graphical browser's graph structures.

This graph contains 79 nodes.

Creating a full view.
Laying out static views.
```

After an RLF library is selected, the RLF GB creates a graphical depiction of the library in a new window, called a "view," or "graph display window." Note that by displaying an RLF library, the Main Menu window is expanded. Therefore, there is still only one RLF GB window being displayed at this point.

The RLF GB graph display window should look similar to Figure 3.

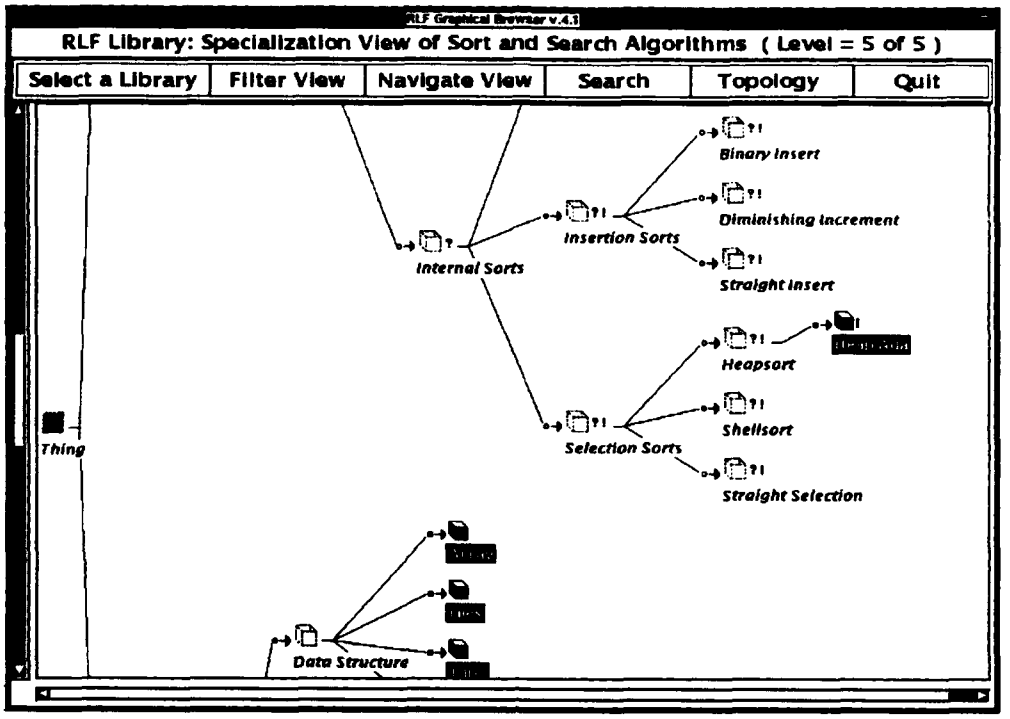


Figure 3: The RLF GB Graph Display Window

If your display looks similar to Figure 3, then the installation was successful. If your display does not look similar to Figure 3, it may be due to configuration problems such as the bitmap files not being found, or inappropriate font names for your site. Or the problem may be something as simple as an inappropriate setting for an environment variable. The first item to check is to ensure that all environment variables have valid settings. The second item to check is error messages—see if the RLF GB application reported any processing errors. Look in the window from which you invoked the RLF GB. Most error messages are reported on "standard out" and so would be found there. If there are any error messages reported, check the appendix in the *RLF User's Manual* to see if the error message is discussed there. If it is, then follow the provided suggestions for solving the problem. If the error message is not listed in the *RLF User's Manual*, and you cannot determine the cause of the problem, then report the problem to the development staff. See Appendix C for problem reporting procedures.

3.2.2 Translating RLF Sample Libraries

The final verification that the RLF installation was successful is to run the Library Model Definition Language (LMDL) and Rule Base Definition Language (RBDL) translators on the sample library model specifications and translate them into browsable RLF libraries. The *RLF Modeler's Manual* explains this process in greater detail, but a brief overview is given here.

This release of the RLF contains sample library model specifications. These specifications can be found in the **models** subdirectory of the **RLFHOME** directory. The sample models contained in this release include the following:

- Ada/Xt (the STARS Ada X bindings)
- Anti-Submarine Warfare (ASW)
- Common Data Model
- Software Technology
- Window Manager Move Domain
- the LMDL Action Submodel

In the **models** subdirectory, each model resides in its own subdirectory. For example, the Window Manager Move Domain model resides in the **window_manager** subdirectory. Within these model subdirectories, C shell scripts are provided that translate the given library model specification into a browsable library. These scripts are named **Build_***library_name*.**csh**, where *library_name* is variable, depending on the particular model subdirectory. To run a library build script, type its name on the command line, as in the following example:

```
% Build_Window_Manager_Lib.csh
```

The script then proceeds to run the LMDL translator on the LMDL spec, and the RBDL translator on the RBDL specs (if any RBDL specs exist). The LMDL and RBDL translators report the progress of the translation with status output statements. A sample output is given below:

```
Creating required sub-directories


Initializing text files


Building LMDL Network from move_domain.lmdl


Parsing LMDL specification.
Parsing completed successfully.

Beginning parse tree attribute evaluation.
Completed parse tree attribute evaluation.
```

```
Creating library model ''Window Managers Move Domain''.
Building library model in memory.

Adding category ''Action Definition''
   Using ''move_domain_concept'' for parent
Adding category ''move_features''
   Using ''move_domain_concept'' for parent
Adding category ''wm_move_components''
   Using ''move_domain_concept'' for parent
Adding category ''wm_product_architecture''
   Using ''move_domain_concept'' for parent
Adding category ''Action Type''
   Using ''Action Definition'' for parent
Adding category ''Action''
   Using ''Action Definition'' for parent
Adding relationship ''has_action_type'' to ''Action''
Adding category ''required_features''
   Using ''move_features'' for parent
Adding category ''optional_features''
   Using ''move_features'' for parent
Adding relationship ''optional_feature'' to ''wm_move_components''
Adding relationship ''required_feature'' to ''wm_move_components''
Adding category ''sunview_move''
   Using ''wm_move_components'' for parent
Adding category ''x10_move''
   Using ''wm_move_components'' for parent
Adding category ''wm_move_operation''
   Using ''wm_product_architecture'' for parent
Adding category ''screen_layout_operation''
   Using ''wm_product_architecture'' for parent
Adding category ''System String''
   Using ''Action Type'' for parent
Adding category ''Display_Action''
   Using ''Action'' for parent
Restricting value of relationship
 ''Display_Action''.''has_action_type''
Adding initial empty subset for ''Display_Action''.''has_action_type''
Adding action tuple of ''Display'' to ''move_domain_concept''
Adding category ''Show_Feature''
   Using ''Action'' for parent
Restricting value of relationship ''Show_Feature''.''has_action_type''
Adding initial empty subset for ''Show_Feature''.''has_action_type''
Adding category ''window_layout''
   Using ''required_features'' for parent
Adding category ''move_input''
   Using ''required_features'' for parent
Adding object ''option_chosen''
```

```
   Using ''optional_features'' for parent
Adding object ''abort_move_operation''
   Using ''optional_features'' for parent
Adding action tuple of ''Show'' to ''abort_move_operation''
Adding object ''move_icon''
   Using ''optional_features'' for parent
Adding action tuple of ''Show'' to ''move_icon''
Adding object ''partially_off_screen''
   Using ''optional_features'' for parent
Adding action tuple of ''Show'' to ''partially_off_screen''
Adding category ''move_resize_feedback''
   Using ''optional_features'' for parent
Adding object ''constrained_move''
   Using ''optional_features'' for parent
Adding action tuple of ''Show'' to ''constrained_move''
Adding category ''move_erasure''
   Using ''optional_features'' for parent
Adding relationship ''when_to_erase'' to ''move_erasure''
Adding object ''expose_after_move''
   Using ''optional_features'' for parent
Adding action tuple of ''Show'' to ''expose_after_move''
Adding object ''sunview_move_description''
   Using ''sunview_move'' for parent
Adding object ''sunview_move_component''
   Using ''sunview_move'' for parent
Adding object ''x10_move_description''
   Using ''x10_move'' for parent
Adding object ''x10_move_component''
   Using ''x10_move'' for parent
Adding object ''overlapped_layout''
   Using ''window_layout'' for parent
Adding category ''tiled_layout''
   Using ''window_layout'' for parent
Adding object ''move_border''
   Using ''move_input'' for parent
Adding object ''move_interior''
   Using ''move_input'' for parent
Adding object ''zap_effect''
   Using ''move_resize_feedback'' for parent
Adding object ''window_configuration''
   Using ''move_resize_feedback'' for parent
Adding category ''interactive_feedback''
   Using ''move_resize_feedback'' for parent
Adding object ''erase_before''
   Using ''move_erasure'' for parent
Adding object ''erase_after''
   Using ''move_erasure'' for parent
Adding object ''tiled_layout_descr''
```

```
   Using ''tiled_layout'' for parent
Adding action tuple of ''Display'' to ''tiled_layout_descr''
Adding object ''tiled_columns''
   Using ''tiled_layout'' for parent
Adding object ''tiled_arbitrary''
   Using ''tiled_layout'' for parent
Adding object ''ghost_feedback''
   Using ''interactive_feedback'' for parent
Adding filler ''erase_after'' to ''sunview_move_component''
Adding filler ''constrained_move'' to ''sunview_move_component''
Adding filler ''ghost_feedback'' to ''sunview_move_component''
Adding filler ''move_border'' to ''sunview_move_component''
Adding filler ''abort_move_operation'' to ''sunview_move_component''
Adding filler ''move_icon'' to ''sunview_move_component''
Adding filler ''overlapped_layout'' to ''sunview_move_component''
Adding object ''opaque_feedback''
   Using ''interactive_feedback'' for parent
Adding filler ''expose_after_move'' to ''x10_move_component''
Adding filler ''erase_after'' to ''x10_move_component''
Adding filler ''window_configuration'' to ''x10_move_component''
Adding filler ''zap_effect'' to ''x10_move_component''
Adding filler ''opaque_feedback'' to ''x10_move_component''
Adding filler ''ghost_feedback'' to ''x10_move_component''
Adding filler ''move_interior'' to ''x10_move_component''
Adding filler ''move_icon'' to ''x10_move_component''
Adding filler ''overlapped_layout'' to ''x10_move_component''

Binding model attributes.

Adding file attribute ''Help'' to ''move_domain_concept''
Adding string attribute ''Display_Action_String'' to
 ''Display_Action''
Adding string attribute ''Show_Feature_String'' to ''Show_Feature''
Adding file attribute ''Contents'' to ''tiled_layout_descr''
Adding file attribute ''Feature_Attribute'' to
 ''abort_move_operation''
Adding file attribute ''Feature_Attribute'' to ''move_icon''
Adding file attribute ''Feature_Attribute'' to
 ''partially_off_screen''
Adding file attribute ''Feature_Attribute'' to ''constrained_move''
Adding file attribute ''Feature_Attribute'' to ''expose_after_move''
Attaching inferencer ''move_domain_concept'' to
 ''move_domain_concept''
Attaching inferencer ''option_move_resize_feedback'' to
 ''move_resize_feedback''
Attaching inferencer ''x10_uwm_move'' to ''x10_move''
Attaching inferencer ''sun_view_move'' to ''sunview_move''
```

```
Saving library model to disk.

Library model created.


Creating Inferencer from move_domain.rbdl

Parsing input.
Parsing completed successfully.
Entering attribute evaluation phase.
Exiting attribute evaluation phase.
Inferencer ''move_domain_concept'' created.

Creating Inferencer from option_move_resize.rbdl

Parsing input.
Parsing completed successfully.
Entering attribute evaluation phase.
Exiting attribute evaluation phase.
Inferencer ''option_move_resize_feedback'' created.

Creating Inferencer from sunview_move.rbdl

Parsing input.
Parsing completed successfully.
Entering attribute evaluation phase.
Exiting attribute evaluation phase.
Inferencer ''sun_view_move'' created.

Creating Inferencer from x10_move.rbdl

Parsing input.
Parsing completed successfully.
Entering attribute evaluation phase.
Exiting attribute evaluation phase.
Inferencer ''x10_uwm_move'' created.
```

After successful translation, the library model is browsable with the RLF Graphical Browser. To verify that the translation was successful, try to browse the model using the procedures described in the previous section.

A Appendix: Customization

Since the RLF GB is an X application, users are able to customize the "look and feel" of the application to a significant degree. The range of customization possible is large, therefore, only the subset of bitmap and font customization options will be discussed in this appendix. For a detailed description of all the possible X customization procedures, see the X references listed in Appendix D, "References."

A.1 X Resources

All X applications have *resources*. These resources consist of items such as bitmaps, fonts, colors, cursors, and windows. All these resources have unique identifiers associated with them for naming purposes; thus, the user or application programmer can set the values of these resources to customize the look and feel of applications.

There is a set of precedence rules that all X applications follow to determine what resources will be applied to a given invocation of the application. Resources that are loaded first will be overridden by those loaded later.

The following list states the rules for setting X application resources in **reverse** order of priority. For example, item 1 will be overridden by item 4, and item 4 will be overridden by item 5.

1. /usr/lib/X11/app-defaults/*Application_Class_Name*
   - the application resource specification file on the host running the client X application
   - this corresponds to the **/usr/lib/X11/app-defaults/RLF_Browser** file for the RLF GB application
2. XAPPLRESDIR environment variable
   - the value of the XAPPLRESDIR environment variable is a directory pathname; it can be set to point to a directory containing a file named *Application_Class_Name*, *e.g.*, **RLF_Browser**, that resides somewhere other than **/usr/lib/X11/app-defaults**
3. Resources loaded into the RESOURCE_MANAGER property of the root window
   - typically, the user arranges to have **xrdb** run from the X initialization file **.xinitrc**
   - if the RESOURCE_MANAGER property is not set, the *resource manager* looks for a **.Xdefaults** file in the user's home directory
4. XENVIRONMENT environment variable

- a complete pathname **including the filename** (different from the XAPPLRESDIR environment variable, which is only the pathname, but does not include the filename)
- if this variable is not defined, then the *resource manager* looks for a file in the user's home directory named `.Xdefaults-`*hostname*

5. Command Line Values

   - specified with the `-xrm` option on the command line
   - values are loaded for that instance of the program only

6. If the application has defined any command line options by passing an options table to the programmatic X call *XtInitialize*, values from the command line will override those specified by any other resource settings.

A default application resource specification file called `RLF_Browser` should exist in `/usr/lib/X11/app-defaults` as a result of the installation procedures described in this document. If not, users should contact the person who installed RLF (e.g. the site's system administrator) to determine where the file was installed. If nothing else is done, the RLF GB will use the resource values in that file.

The most convenient means of customizing the RLF GB is to make your own copy of the `RLF_Browser` file and set XAPPLRESDIR to point to its *directory* location. Then you can edit the `RLF_Browser` file and change any resources specified in that file, such as bitmaps, fonts, window sizes and window placement to suit individual needs.

A.2 Bitmaps

The RLF GB, Version 4.1, uses the Motif widget set, therefore, Motif configuration standards and conventions must be used. This has a significant impact on how resources are set, particularly in relation to earlier versions of the RLF GB. The result is that most of the methods for setting resources are different from pre-4.1 versions.

The pathname to the `bitmaps` directory is no longer specified in the old `Browser` file, or the new `RLF_Browser` file. In Motif applications, the `bitmaps` directory must be a subdirectory in the resource file's pathname. For example if you set the XAPPLRESDIR environment variable to `/usr/lib/X11/rlfgb`, then the RLF GB looks for the `bitmaps` directory to be at `/usr/lib/X11/rlfgb/bitmaps`. This directory must contain the bitmaps used by the RLF GB.

You can change XAPPLRESDIR to point to a different directory, and therefore a different `bitmaps` directory, to use bitmap files that you have created or customized. If the pathname is not specified correctly, the RLF GB will not be able to find its bitmaps, and blank space will be displayed instead. It is important that the resource file's directory pathname be specified correctly and for the `bitmaps` directory to be located in that directory as a subdirectory or else the RLF GB graphic display will be aesthetically less pleasing. The bitmaps directory included in the RLF release may be used directly unless use of alternative bitmaps is desired, in which case you may create a `bitmaps` directory in the location of your choice, containing the desired bitmaps.

The specific bitmaps used by the RLF GB to represent displayed objects can be specified in the `RLF_Browser` file. The different types of nodes in an RLF GB graph view each have a bitmap defined for it. For example, all category nodes that do not have any actions or inferencers attached use the bitmap file specified by the following line in the `RLF_Browser` file:

```
vshell*scr_window*node_CATEGORY_KIND.labelPixmap:  box_m.xbm
```

while all object nodes that do not have any actions or inferencers attached use the bitmap file specified by the following line in the `RLF_Browser` file:

```
vshell*scr_window*node_OBJECT_KIND.labelPixmap:  cube_m.xbm
```

All category nodes could be changed to display a different bitmap by editing the above line in the `RLF_Browser` file and specifying a different value for the bitmap file. For example, to change the category nodes to a bitmap you created and placed in a file called `my_bitmap.xbm`, you would set the value in the `RLF_Browser` resource file to the following:

```
vshell*scr_window*node_CATEGORY_KIND.labelPixmap:  my_bitmap.xbm
```

Alternatively, you may specify an absolute pathname for any particular bitmap. For example, you could use bitmap files from your home directory, as in the following example:

```
vshell*scr_window*node_CATEGORY_KIND.labelPixmap:  /home/my_name/my_bitmap.xbm
```

A.3 Fonts

Fonts used by the RLF GB application can be changed in the same manner as bitmaps. There are lines in the `RLF_Browser` file that specify what fonts to use for particular textual objects in the RLF GB display. For example, the font used for Category concept names in the graph display is specified in the following line in the `RLF_Browser` file:

```
vshell*scr_window*node_label_CATEGORY_KIND*fontList: -b&h-lucida-bold-i-*-*-17-*-*-*-*-*-*-*
```

and similarly, this font name could be modified to suit varying needs. For instance, the display of very large graphs sometimes necessitates the use of a smaller font (and/or smaller bitmaps) so that the graph display will not exceed the capacity of X to display it. Or sometimes very large fonts and bitmaps are desired for demonstration purposes. These types of customizations and many others are most easily accomplished by modifying the `RLF_Browser` file and setting the XAPPLRESDIR environment variable appropriately.

A.4 Invoking the Browser from a Shell Script

The C shell script `RLF_GB` checks the appropriate environment variables to determine as best it can whether they are valid, and then invokes the RLF Graphical Browser. This method of invocation is recommended for novice and beginning RLF users. The `RLF_GB` script is found in the `unix/bin` or `pcte/bin` directory of this software release, along with the `Graphical_Browser` executable.

A.5 Command Line Arguments

Another way of customizing the look of the RLF GB is to set X resources via the command line. Various X resources can be specified by invoking the X application with the appropriate command line arguments. As can be seen in the list of precedence rules, this method will override any other previous settings.

An example of invoking the RLF GB with command line arguments is as follows:

For an extra-large library, use a small font and small bitmaps:

```
Graphical_Browser \
 -xrm "vshell*scr_window*node_label_CATEGORY_KIND*fontList: -b&h-*-*-r-*-*-10-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_CATEGORY_W_ACTION*fontList: -b&h-*-*-r-*-*-10-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_CATEGORY_W_ADVICE*fontList: -b&h-*-*-r-*-*-10-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_CATEGORY_W_ADVICE_AND_ACTION*fontList: -b&h-*-*-r-*-*-10-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_OBJECT_KIND*fontList: -b&h-*-*-r-*-*-10-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_OBJECT_W_ACTION*fontList: -b&h-*-*-r-*-*-10-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_OBJECT_W_ADVICE*fontList: -b&h-*-*-r-*-*-10-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_OBJECT_W_ADVICE_AND_ACTION*fontList: -b&h-*-*-r-*-*-10-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_CATEGORY_KIND.labelPixmap: box_xs.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_KIND.reverseLabelPixmap: box_rev_xs.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ACTION.labelPixmap: box_A_xs.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ACTION.reverseLabelPixmap: box_A_rev_xs.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ADVICE.labelPixmap: box_I_xs.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ADVICE.reverseLabelPixmap: box_I_rev_xs.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ADVICE_AND_ACTION.labelPixmap: box_AI_xs.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ADVICE_AND_ACTION.reverseLabelPixmap: box_AI_rev_xs.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_KIND.labelPixmap: cube_xs.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_KIND.reverseLabelPixmap: cube_rev_xs.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ACTION.labelPixmap: cube_A_xs.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ACTION.reverseLabelPixmap: cube_A_rev_xs.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ADVICE.labelPixmap: cube_I_xs.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ADVICE.reverseLabelPixmap: cube_I_rev_xs.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ADVICE_AND_ACTION.labelPixmap: cube_AI_xs.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ADVICE_AND_ACTION.reverseLabelPixmap: cube_AI_rev_xs.xbm"
```

For a demonstration, you might want to use a large font and large bitmaps:

```
Graphical_Browser \
 -xrm "vshell*scr_window*node_label_CATEGORY_KIND*fontList: -b&h-*-*-r-*-*-20-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_CATEGORY_W_ACTION*fontList: -b&h-*-*-r-*-*-20-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_CATEGORY_W_ADVICE*fontList: -b&h-*-*-r-*-*-20-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_CATEGORY_W_ADVICE_AND_ACTION*fontList: -b&h-*-*-r-*-*-20-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_OBJECT_KIND*fontList: -b&h-*-*-r-*-*-20-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_OBJECT_W_ACTION*fontList: -b&h-*-*-r-*-*-20-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_OBJECT_W_ADVICE*fontList: -b&h-*-*-r-*-*-20-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_label_OBJECT_W_ADVICE_AND_ACTION*fontList: -b&h-*-*-r-*-*-20-*-*-*-*-*-*-*" \
 -xrm "vshell*scr_window*node_CATEGORY_KIND.labelPixmap: box_m.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_KIND.reverseLabelPixmap: box_rev_m.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ACTION.labelPixmap: box_A_m.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ACTION.reverseLabelPixmap: box_A_rev_m.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ADVICE.labelPixmap: box_I_m.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ADVICE.reverseLabelPixmap: box_I_rev_m.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ADVICE_AND_ACTION.labelPixmap: box_AI_m.xbm" \
 -xrm "vshell*scr_window*node_CATEGORY_W_ADVICE_AND_ACTION.reverseLabelPixmap: box_AI_rev_m.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_KIND.labelPixmap: cube_m.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_KIND.reverseLabelPixmap: cube_rev_m.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ACTION.labelPixmap: cube_A_m.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ACTION.reverseLabelPixmap: cube_A_rev_m.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ADVICE.labelPixmap: cube_I_m.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ADVICE.reverseLabelPixmap: cube_I_rev_m.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ADVICE_AND_ACTION.labelPixmap: cube_AI_m.xbm" \
 -xrm "vshell*scr_window*node_OBJECT_W_ADVICE_AND_ACTION.reverseLabelPixmap: cube_AI_rev_m.xbm"
```

The above command-line invocations of the RLF GB with command line arguments specify the font and bitmaps of the graph view window that is displayed. These command line arguments override any other resource settings that may been specified previously. Editing

startup scripts like these, or typing new command line arguments manually, are more examples of the many options available for customizing X applications such as the RLF GB. Other X resource customizations, such as window size and placement, can also be accomplished using the command line argument method.

For further possibilities of customizing X resources, consult the X Window System reference given in Appendix D, References.

B Appendix: PCTE

In most respects, the PCTE version of this delivery of RLF operates in the same manner as the UNIX version. However, there are differences, and this appendix presents the differences in the PCTE and UNIX versions of RLF and some conventions which can be used to increase portability between versions. This appendix assumes knowledge of PCTE, the Emeraude PCTE product, and the *esh* encapsulated shell. A major assumption for this release of the PCTE RLF is that you install the software into UNIX, then run *esh* scripts in PCTE to install RLF libraries into the PCTE object base, then run the RLF GB from UNIX (since it's not encapsulated - you just use the absolute pathname), and the PCTE version of the RLF GB then accesses the objects in the PCTE object base. For a more detailed discussion of library modeling issues with the PCTE version of the RLF, see the *RLF Modeler's Manual.*

B.1 Installing the PCTE Version of RLF

The installation procedures for the PCTE version of RLF are similar to the installation procedures for the UNIX version of RLF. Some of the differences have been noted in the preceding sections of this manual, and some of the differences are obvious. For example, the interactive installation script`Install_RLF_pcte_bin` is used instead of `Install_RLF_bin`.

The PCTE installation proceeds the same as the UNIX installation, except no RLF libraries are automatically created for the PCTE version. You must start *esh* (encapsulated shell) scripts from inside the PCTE environment to create RLF libraries in the PCTE object base. The procedures for running these scripts are provided in the next section.

B.2 Verifying the PCTE RLF Installation

The procedures for verifying the PCTE RLF installation are similar to the UNIX version, but differ because they ultimately must be performed from within the PCTE environment.

After the installation of the PCTE RLF system has completed, you should be able to determine that the `pcte/bin` subdirectory of the `RLFHOME` directory contains the following executable programs by issuing an *ls* command:

```
% ls -a $RLFHOME/pcte/bin
```

The *ls -a* command should produce results similar to the following:

```
.rlfrc
Graphical_Browser
Library_Manager
Lmdl
```

```
RLF_Browser
RLF_GB
Rbdl
bitmaps (a directory)
pcte.profile
pcte_install
```

B.2.1 Starting the X Window System

An X server must be running on your local workstation for the RLF GB to be successfully invoked. However, it is beyond the scope of this installation guide to describe how to install X, or how to start the X server on a workstation. Consult your local system administrator for help.

B.2.2 Installing the *esh* Init File

Before invoking the RLF GB, you may first want to install the `.profile` file into the PCTE object base. The `.profile` is an initialization file that is read by the *esh* when it is invoked. This step is optional; the `.profile` file is used for convenience—it gathers frequently used commands into a file and performs those commands automatically upon shell invocation.

The `.profile` file supplied with this release contains the necessary commands to set the appropriate environment variables. However, the pathnames provided should be tailored for your site by editing the `.profile` file with the text editor of your choice:

```
% editor $RLFHOME/pcte/bin/pcte.profile
```

The `.profile` file in the current release looks similar to the following; note that the *esh* is an encapsulation of the Bourne shell (*sh*), and therefore uses Bourne shell syntax:

```
XAPPLRESDIR=$RLFHOME/pcte/bin; export XAPPLRESDIR
PATH="/rlf.tools:$PATH; export PATH
RLF_LIBRARIES="/Instances4.e; export RLF_LIBRARIES
RLF_PAGER=/usr/local/bin/less; export RLF_PAGER
```

If any of the above pathnames are not appropriate for your site, then change them to reflect more suitable values. There are two environment variables that must be set before the RLF GB can be run successfully. They are as follows:

- DISPLAY

- RLF_LIBRARIES

You can edit the `.profile` file so that these variables are automatically set every time you invoke a new *esh* shell. The DISPLAY variable is not set in the above sample `.profile` file, but it could be set there if desired. Note that environment variables *are* inherited by the *esh* from its parent process.

The XAPPLRESDIR environment variable contains the location of the `RLF_Browser` X resource file. The `bitmaps` directory should exist as a subdirectory to this location.

The RLF_PAGER environment variable must be set to the pathname of a text pager. This is the text pager that the RLF GB will invoke inside an *xterm* window to display text.

The RLF_LIBRARIES environment variable is used by the RLF to determine the location of the RLF libraries to be read. An RLF library must be created before the RLF GB can be used to browse that library.

B.2.3 Starting the PCTE Server

Consult the *Emeraude V12 System Administration Guide* for information on how to start the PCTE server.

B.2.4 Logging into PCTE

The "standard" method for logging into PCTE is sufficient for using the RLF GB. Consult the *Emeraude V12 System Administration* guide for details.

Before logging in to PCTE, set the RLFHOME environment variable, if it hasn't already been set. The value of the RLFHOME environment variable should be the same as that set in the `Build_RLF.var` file. The pathname is probably the same as the location where the release was extracted from the transfer media.

The RLFHOME environment variable is used for convenience; you could type the entire absolute pathname instead, but that can be cumbersome.

```
% setenv RLFHOME pathname
```

Assuming your command search path is set properly to find the Emeraude PCTE commands, and assuming you have been set up as valid PCTE user, you should be able to issue the *log* command as follows:

```
% log
```

This should invoke an *esh* shell with a prompt similar to the following:

```
esh$
```

B.2.5 Creating RLF Libraries in the PCTE Object Base

After the `.profile` file has been tailored to your satisfaction it must be installed in the PCTE object base before it can have any efffect. The UNIX file is copied into the object base with the PCTE *object_copy* command as in the following example:

```
esh$ obj_copy -c $RLFHOME/pcte/bin/pcte.profile $PCTE_HOME/.profile
```

Before the RLF GB can be successfully invoked in the PCTE environment, an RLF library must be installed into a PCTE object base. To accomplish this, the following procedures may be used.

Any of the *esh* scripts in the `models` subdirectory of the RLF release may be used to create the associated RLF library in the PCTE object base. The following examples show the creation of the "Animals" RLF library.

Invoke the *esh* script using the full UNIX pathname to the script, with an argument of the pathname to the RLFHOME location:

```
esh$ $RLFHOME/models/animals/Build_Animals_Lib.esh $RLFHOME
```

The script `Build_Animals_Lib.esh` copies the necessary files from UNIX into the PCTE object base and executes the `Lmdl` translator on the LMDL script.

B.2.6 Installing the RLF Tools into the PCTE Object Base (Optional)

You may want to install the RLF tools into your PCTE object base instead of invoking them from UNIX. This is completely optional. This section may be ignored if you do not want to do this, since the tools can be successfully invoked from UNIX. However, if you do want to install the RLF tools into the object base, then follow the procedures described in this section.

There is a script called `$RLFHOME/pcte/bin/pcte_install` that installs an `rlf.tools` toolset directory in the location specified by `$PCTE_HOME`. You may not be able to create toolset objects everywhere, but you should be able to install it in the PCTE home directory, which is the default. Pass the UNIX path of the location where the RLF release was extracted from the archive media (usually this is also specified by `$RLFHOME`) as a parameter to this script:

OPTIONAL:

```
esh$ $RLFHOME/pcte/bin/pcte_install $RLFHOME
```

B.2.7 Invoking the PCTE RLF Graphical Browser

After an RLF library has been installed in the PCTE object base, the RLF `Graphical_Browser` may be invoked. To invoke the RLF GB from within the PCTE environment, use the full UNIX pathname as in the following example:

```
esh$ $RLFHOME/pcte/bin/Graphical_Browser
```

B.3 File Naming Restrictions

The Emeraude implementation of PCTE places restrictions on the length of object names and makes assumptions about the use of '.' in object names. The names of files containing assets which are available in an RLF reuse library are restricted to 32 characters in length when using PCTE. These are the files that reside beneath the `Text` subdirectory of any directory where RLF libraries have been constructed. Additionally, the names of files containing reusable assets in the library should not contain the '.' character, since this indicates a special meaning to the Emeraude implementation of PCTE. The convention established by this version of RLF for PCTE is to replace any '.' characters in file names with the underscore character, '_'. An exception to this convention is the .rlfrc start-up file, which the PCTE version of RLF will look for as an entity named `rlfrc.e`.

To increase the similarity in the way libraries are represented in the UNIX and PCTE versions, and to ease transition between versions, the preferred link type of every object in or beneath the directory object where the library was built must be set to "`.e`". This includes files representing a library's assets and any action scripts which might appear below the `Text` directory. The preferred link type of the directory object indicated by the environment variable, RLF_LIBRARIES, also needs to be "`.e`" so that its subdirectories can be traversed easily.

Library representations built with the PCTE version of RLF also require a directory object named `rlf_tools` to be a first-level subdirectory of the directory object where the library is built. This directory object must also contain two tools named `ascii_file.tool` and `displ_attr.tool`. These tools are required for RLF's default actions to operate correctly.

For examples of library model construction for the PCTE version of RLF, examine the "`.esh`" versions of the build scripts for the example libraries delivered with the RLF. These scripts are found in each subdirectory of the `models` subdirectory of an RLF installation. These scripts can be modified and reused to help automate the procedures required to build an RLF reuse library with the PCTE version.

C Appendix: Reporting Problems

Like most software, especially that of a prototypical nature, the RLF GB may contain unknown bugs (along with some known bugs). If you encounter any problems with this software, or have any suggestions for enhancements, you are encouraged to report them to STARS personnel. The *Version Description Document* included with this release provides instructions for doing this, including information on available Internet electronic mail addresses. Also included in the distribution is a problem report form for formally submitting problem reports.

D Appendix: References

This section lists a number of resources that are available for further information. The topics are relevant to the use of the RLF GB and include the RLF, the RGB, the X Window System, Ada, and PCTE references.

For more detailed information on the RLF, see the following documents:

Δ • *RLF Binary Release Installation Guide, Version 4.1*, STARS-UC-05156/012/00; March, 1993.

Δ • *RLF User's Manual, Version 4.1*, STARS-UC/05156/013/00); March, 1993.

Δ • *RLF Administrator's Manual, Version 4.1*, STARS-UC-05156/017/00; March, 1993.

Δ • *RLF Modeler's Manual, Version 4.1*, STARS-UC-05156/011/00; March, 1993.

Δ •

RLF Binary Release Version Description Document, Version 4.1, STARS-UC-05156/016/00; March, 1993.

Δ • *RLF User Tutorial, Version 4.1*, STARS-UC-05156/018/00; March, 1993.

Δ • *RLF Administrator Tutorial, Version 4.1*, STARS-UC-05156/019/00; March, 1993.

Δ • *RLF Modeler Tutorial, Version 4.1*, STARS-UC-05156/020/00; March, 1993.

For more detailed information on the RGB, see the following documents:

- *RGB 1.0 Version Description Document (VDD)*, STARS-US-020401/001/00
- *RGB 1.0 User's Manual*, STARS-US-020401/002/00

For more detailed information on using X and the *twm* window manager, see the following book:

- *X Window System User's Guide for X11 R3 and R4, Third Edition*; Quercia, Valerie, and O'Reilly, Tim; O'Reilly & Associates, Inc.; Sebastapol, CA; 95472; May 1990.

For more detailed information on SA-Motif see the following documents:

- *Ada/Motif Release 1.1, The Complete Ada Binding for X11R4 and OSF/Motif 1.1, Users Manual*, Systems Engineering Research Corporation (SERC), Mountain View, CA; Sept. 28, 1992.

For more detailed information on Sun Ada see the following documents:

- *Sun Ada User's Guide*, March, 1992
- *SPARCworks/Ada User's Guide*, March, 1992

For more detailed information on Emeraude PCTE see the following documents:

- *The Emeraude Environment*, GIE Emeraude, 1992; PC6A1, 68, route de Versailles, 788430 Louveciennnes, FRANCE.